Deep Learning for Semantic Segmentation of Airplane Hyperspectral Imaging

A Degree Thesis submitted to
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya

In partial fulfillment of the requirements for the
Bachelor's degree in Telecommunications Technologies and Services Engineering

Author
Mar Balibrea Rull

Advisors
Verónica Vilaplana Besler
Luis Fernando Salgueiro

Barcelona, June 2019

Deep Learning for Semantic Segmentation of Airplane Hyperspectral Imaging

Abstract

Given their success, both qualitative and quantitative, Deep Neural Networks have been used to approach classification and segmentation problems for images, especially during these last few years where it has been possible to design computers with sufficient capacity to make quick and efficient experiments.

In this work, we will study the use of two Convolutional Neural Networks (CNNs) to segment the ground of a land section of Maspalomas' Park using an image taken by the flight of an airplane.

The comparison will be made in terms of computational cost, complexity and results that will be obtained while testing different algorithms, loss functions or optimizers and also while tuning some other parameters. The results will also be compared with a past work [12] done with the same dataset but another methodology (SVM).

Aprenentatge Profund per Segmentació Semàntica d'Imatges Hiperespectrals d'Avió

Resum

Tenint en compte el seu èxit, tant qualitatiu com quantitatiu, s'han utilitzat Xarxes Neuronals Profundes per abordar problemes de classificació i segmentació d'imatges, especialment durant aquests últims anys on s'han pogut dissenyar ordinadors amb capacitat suficient per fer experiments ràpids i eficients.

En aquest treball, estudiarem l'ús de dues xarxes neuronals convolucionals (CNNs) per segmentar el sòl d'una secció del Parc de Maspalomas mitjançant una imatge presa amb el vol d'un avió.

La comparació es farà en termes de cost computacional, complexitat i resultats que s'obtindran en provar diferents algorismes, funcions de pèrdua o optimitzadors i, a més, ajustant alguns altres paràmetres. Els resultats també es compararan amb un treball anterior realitzat [12] amb el mateix conjunt de dades, però amb una altra metodologia (SVM).

Aprendizaje Profundo para Segmentación Semantica de Imágenes Hiperespectrales de Avión

Resumen

Teniendo en cuenta su éxito, tanto cualitativo como cuantitativo, se han utilizado Redes Neuronales Profundas para abordar problemas de clasificación y segmentación de imágenes, especialmente durante estos últimos años donde se han podido diseñar ordenadores con capacidad suficiente para hacer experimentos rápidos y eficientes.

En este trabajo, estudiaremos el uso de dos redes neuronales convolucionales (CNNs) para segmentar el suelo de una sección del Parque de Maspalomas mediante una imagen tomada por el vuelo de un avión.

La comparación se hará en términos de coste computacional, complejidad y resultados que se obtendrán en probar diferentes algoritmos, funciones de pérdida o optimizadores y, además, ajustando algunos otros parámetros. Los resultados también se compararán con un trabajo anterior [12] realizado con el mismo conjunto de datos, pero con otra metodología (SVM).

Acknowledgements

First, I would like to thank my advisor Verónica Vilaplana for her guidance, knowledge and help orienting the project by thinking of different interesting perspectives to consider.
Also, I would like to acknowledge Luis Fernando Salgueiro for his daily support and advice with overall problems of the development.
Last but not least, I would like to show appreciation to my close friends and family to encourage me throughout my educational path.

Revision history and approval record

Revision	Date	Purpose
0	22/02/2019	Document creation
1	18/06/2019	Document revision
2	25/06/2019	Document revision

DOCUMENT DISTRIBUTION LIST

Name	E-mail
Mar Balibrea Rull	mar.balibrea@estudiant.upc.edu
Verónica Vilaplana Besler	veronica.vilaplana@upc.edu
Luis Fernando Salgueiro Romero	luis.fernando.salgueiro@upc.edu

WRITTEN BY:		REVIEWED AND APPROVED BY:	
Date	25/06/2019	Date	25/06/2019
Name	Mar Balibrea Rull	Name	Verónica Vilaplana Besler
Position	Project Author	Position	Project Supervisor

Table of Contents

List of Figures

List of Tables

Glossary

DL	Deep Learning
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent
ReLU	Rectified Linear Unit
HSI	Hyperspectral Imaging

1

Introduction

1.1 Statement of purpose

The Institute of Oceanography and Global Change (IOCAG) in collaboration with UPC proposed to use Deep Learning techniques to approach the problem of segmenting different types of land in Maspalomas.

Because of this interest, this problem was undertaken last year in [12] using the SVM technique. Besides this result, there are not other complete maps of this area. The fact of having a labelled map of a park is interesting to have in terms of security and knowledge about land distribution and species present. However, the task of creating these types of maps is, at some grade of detail, difficult and has to be done by someone professional. Overall, is also very time consuming, especially considering that it has more than 4 squared meters of extension.

Based on the above-mentioned motivation, the main objectives of this project are:

- Study the state of the art of Deep Learning for classification and segmentation techniques applied to hyperspectral imagery;
- Analize one of the frameworks used for CNN systems;
- Develop and compare two CNNs that successfully classify and segment types of land in Maspalomas with hyperspectral high resolution airplane imagery;

With the purpose of quantifying these objectives, the proposed project requirements are the following:

- Have a solid and extensive state of the art knowledge about DL for segmentation of hyperspectral imaging to correctly support and background this project;
- Implement a DL model that successfully segments types of land of the input images;

– Compare the result with the ones obtained using techniques based on non-DL methods;

1.2 Technical Remarks

This project was carried out at the Image Processing Group (GPI) within the department of Signal Theory and Communications (TSC) of the Technical University of Catalonia (UPC).

This classification problem was already approached in a Master's Thesis carried out at IOCAG [12] using Support-Vector Machine (SVM).

For the two NN architectures used, they were based on [10], which are developed after the ideas exhibited in [14] and [4] respectively.

The project has been developed in Python using Pytorch [1] framework.

Other software used was the visualizing and processing of hyperspectral images: both ENVI and QGIS were used to open and pre-analyze all the data. ENVI was also used to generate a a segmentation map with SVM for comparison purposes.

1.3 Work organization

The organization for this work was made with barely any experience with developing a project like this. This reflects on the Time Plan changes massively, since the Work Packages with more changes are the ones dedicated to development (see more information in Appendix A: Organization).

However, there was also a factor that made timings change, and that is the labelling. Initially, the image given was going to be labelled manually, but that was very time consuming, so after thinking of ways to avoid not-automatic segmentation for the ground truth, the option of using the result of [12] was brought up. However, that way we wouldn't be able to compare results. Finally, what was used were the manual labels used in [12] to execute his segmentation algorithm.

In Figure 1.1, we can see the cells with a '+' or a '-' indicate whether they were added or removed from the original Time Plan respectively.

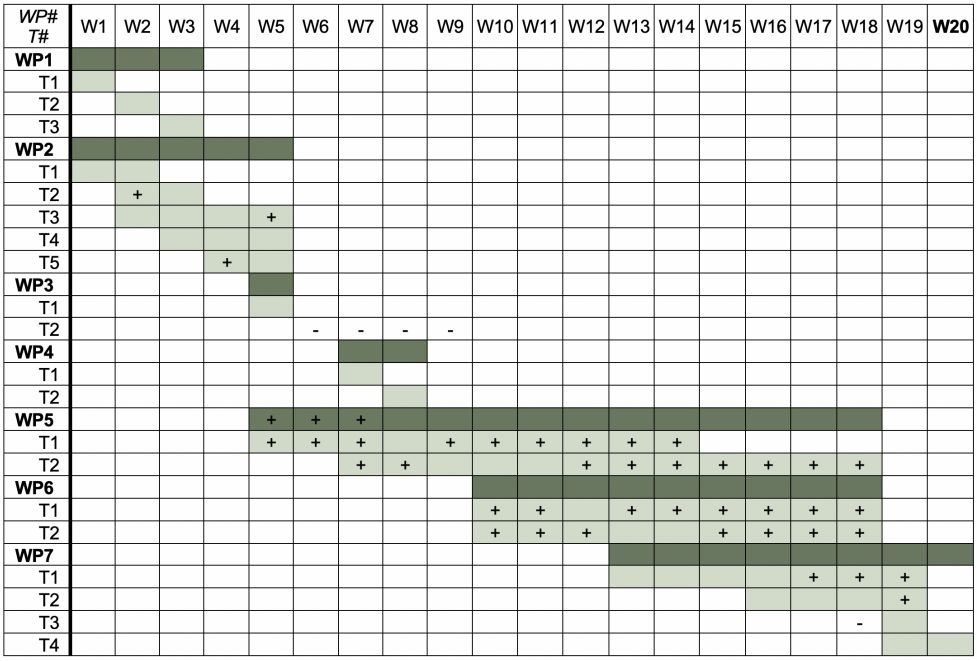

FIGURE 1.1: Updated Gantt diagram

For reference,

- W1 is the week starting the 18th of February 2019
- W3 is the first week starting on March, the 4th of March 2019
- W7 is the first week starting on April, the 1st of April 2019
- W12 is the first week starting on May, the 6th of May 2019
- W16 is the first week starting on June, the 3rd of June 2019
- W20 is the first week starting on July, the 1st of July 2019

2

State of the art

In the last few years, it has been a broad objective to use satellite imagery as the source for fully automated analysis methods, which calls for new ways to obtain reliable information.

Deep Learning is one option that leverages the huge computing power of current machines to perform human-like reasoning and extract features. The interest of the remote sensing community towards Deep Learning methods is growing fast, and many architectures have been proposed these past years, often with an outstanding performance.

In this work, we want to use these networks for segmentation. Segmentation is the procedure of dividing an image in exclusive regions. Each and every pixel of an image can go through a NN and be predicted to belong to one of the classes with some probability. In our case, the segmentation is called semantic because each class has a semantic description like "tree" or "water".

The NNs used will be Convolutional Neural Networks. Like every deep neural network, they have multiple neurons that perform a simple dot product operation between its weights and biases (these being the parameters that are actually *trained*) and the input, and before going to the next one, a non-linear function denominated activation function is applied. This whole process is called the forward pass.

At the end, some type of cost function is performed to then proceed to the backward pass to adapt the weights and biases of each neuron. Figure 2.1 from [17] graphically shows how features can be learned throughout a CNN.

CNNs have been widely used for classification of hyperspectral imaging with different configurations. One of the first convolutional architectures used was the Fully Convolutional Network (FCN). Since 2012 [8], deep convolutional neural networks (CNNs) have dominated computer vision due to their relatively high accuracy at image recognition tasks. For example, in [18], they use FCN with "atrous" convolutions and a multiscale structure to increase the feature density and preserve the resolution; or [16], that uses a FCN for semantic labelling. Other more advanced

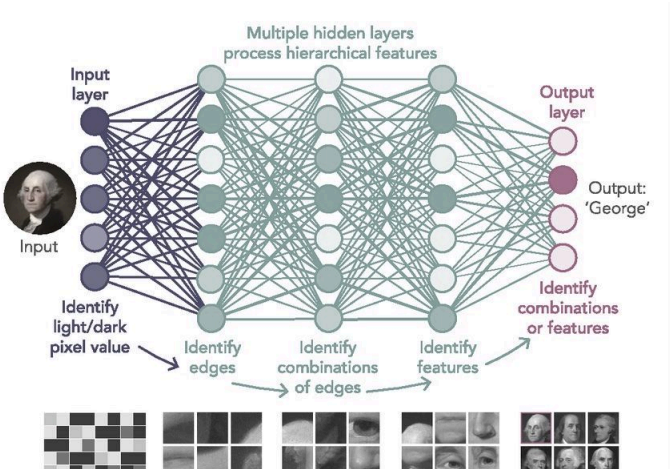

FIGURE 2.1: General CNN architecture schema

techniques are added in [2] (Markov Random Field and Conditional Random Field) to take into account the dependencies between regions or [5], that uses Attribute Profiles to better predict the class labels for HSI pixels.

Another used technique are the Residual Networks (ResNets), that are meant to facilitate the training of deep networks ([6]) by using skip connections or short-cuts to jump over some layers. This idea of passing information to other layers is also adopted by [14] or other variants like [11] or [4], that take into account information shared (correlation) from band to band.

Several papers use more complex architectures by mixing different popular types of nets with HSI datasets. For example [13], that use it to locate different types of objects, or [9], that mixes residual learning with FCN to segment images.

U-Net based papers [14, 4, 11] jointly with [15], that makes a comparison of various 2D and 3D CNNs for HSI classification, started the idea for this work: the goal was to have two fairly simple architectures to compare, without a lot of parameters to learn, that performed well on our dataset. One interesting way to compare them was based on the spectral bands. How did another dimension for convolutions affected performance?

That is why at the end we chose to compare U-Net and 3D U-Net, because the architecture is similar, not very complex, except one has more parameters regarding the depth (spectral bands) of the input. The decision between the 3D U-Net and V-Net was due to complexity (V-Net has residual learning).

The loss function proposed is cross-entropy, in particular weighted cross-entropy, to prevent a strongly biased estimation toward the class with more samples. In [3] is proposed the idea of using weights inversely proportional to the class frequencies.

3

Methodology

In this section, there will be details about pre-processing the data, details of the training of the networks and the evaluation. Finally, a brief comment about the software and hardware specifications.

3.1 System input

3.1.1 Division of data for training and testing

A technique used to decide which data is used for training and for testing is called Stratified sampling. Its general definition is to obtain a fragment of something that best represents the entire collection being studied. In our case, it applies to how the pixels of the dataset are decided to count for training and testing.

Considering N_c, $c = 1 \dots N_{classes}$ samples of each class in the dataset and percentage of samples for training and testing, stratified random sampling assures that the classes are equally distributed as the original.

3.1.2 Data augmentation

At the beginning of each epoch, a simple augmentation algorithm is performed. Basically, with a 50% probability, each image can be flipped vertically or horizontally.

3.2 System architecture

3.2.1 Network architecture

Two different architectures have been studied in order to discern which one combined with other configurations has the best behavior: The U-Net [14] and the 3D U-Net

[4]. Both of them are considered CNNs.

Particularly, Convolutional Networks normally have three types of layers:

- Convolution layer: convolution between the image and filters to train that will create an activation map. This convolution comes with various hyperparameters to configure and they are important because as the network gets trained, the coefficients for these filters will activate in different scenarios.
- Pooling layer: used for downsampling with a "filter" to decide which samples you keep.
- Fully-Connected layer: used to apply all the features outputed by the net and organize them to make a decision.

These are used both for classification and segmentation. For the first one, the network wants to recognize characteristics (for example objects); in the second one, the network also needs to localize them. Hence it is the same as the classification problem but at a pixel level.

U-Net

The U-Net is a CNN used for segmentation, so that every pixel can be classified into one class. This architecture follows first the habitual path for a CNN (various layers of convolutions and poolings to reduce the image size) but then, before performing a 1x1 convolution to shape the feature map to the desired output size, it expanses with more convolutions. Also, in the upsampling part, results from the contracting part are concatenated to get to a more accurate result. See Figure 3.1.

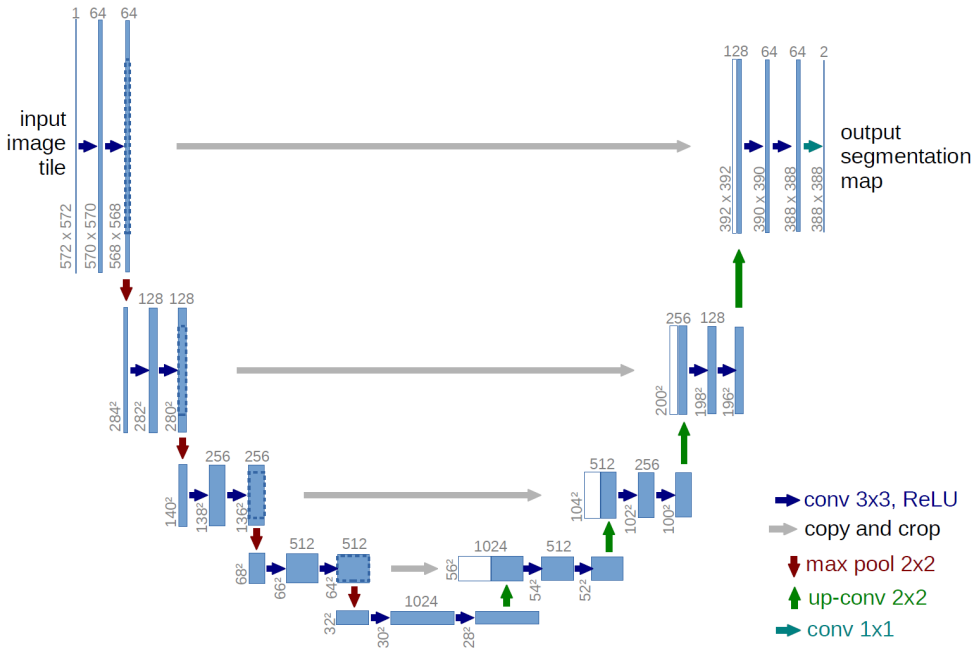

FIGURE 3.1: U-Net architecture

To reduce the number of parameters to learn, we decided to remove the deepest layers. The final architecture can be explained like:

1. Initial *contracting* layer has two iterations of:
   - 2D $3x3$ convolution with 64 channels as output
     (Also a stride of 1, zero padding of 1, dilatation of 1 and not weighted)
   - 2D batch normalization
   - ReLU as activation function
2. Second *contracting* layer has first a 2D max pooling and then the same structure as the previous one with 128 channels as the convolution output
3. Third *contracting* layer has the same structure as the previous one
4. Initial *expanding* layer has first a bilinear 2D interpolation and then two iterations of:
   - 2D $3x3$ convolution with 64 channels as output
     (Also a stride of 1, zero padding of 1, dilatation of 1 and not weighted)
   - 2D batch normalization
   - ReLU as activation function
5. Second *expanding* layer has the same structure as the previous one

6. Final convolution layer has a only a 2D $1x1$ convolution with 13 output channels ($N_{classes}$)

7. Before outputing the final feature map, there is a logarithmic Softmax to to map the non-normalized output to a probability distribution

This way, if the initial image has a shape of $MxNxB$, the output will be of shape $MxNxN_{classes}$.

3D U-Net

This other network is also a CNN used for volumetric segmentation. Its structure is very similar to U-Net, as we can see in Figure 3.2

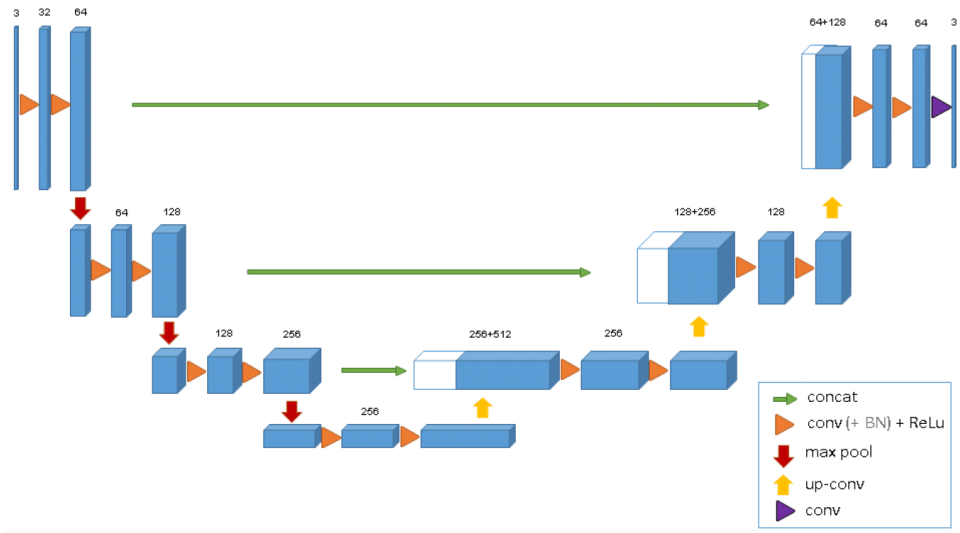

FIGURE 3.2: 3D U-Net architecture

This time, all the operations are 3D instead of 2D (convolution, batch normalization, max pooling and interpolation). This takes into account more information between bands but it obviously increases complexity.

3.2.2 Other functions

While talking about the system architecture, there were some methods mentioned but not explained. As to briefly comment some of them, here is a concise description:

– Batch normalization

– ReLU as activation function

As mentioned, activation functions are used to add non-linearity. Without them, the network would not really learn. There are many functions that can be used but the Rectified Linear Unit (ReLU) is the most common because for positive values (the majority) there is low saturation and the gradient is not close to 0 so it doesn't make the learning slow. See in Figure 3.3.

$$ReLU(x) = max(0, x) \tag{3.1}$$

– Pooling

This is used in convolutional layers of both architectures mentioned. Basically, it is like the technique used in downsampling to obtain some values of the data, but the value you keep can be based not just on its position, but its value. For example, in max pooling, the maximum value is the one that stays.

– Logarithmic Softmax

Softmax is actually another activation function very much used at the end of a network for classification. It gives meaning to the scores converting them to a probability distribution. See in Figure 3.4.

$$LogSigmoid(x) = log(\frac{1}{1 + exp(-x)}) \tag{3.2}$$

– Upsampling

This is used in the *expanding* layers to perform the "inverse" operation of the pooling. Various types of interpolation can be applied, even other methods like transposed convolution. However, in this case bilinear interpolation is used.

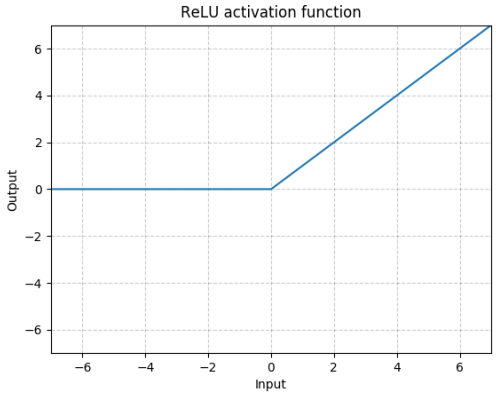

FIGURE 3.3: ReLU

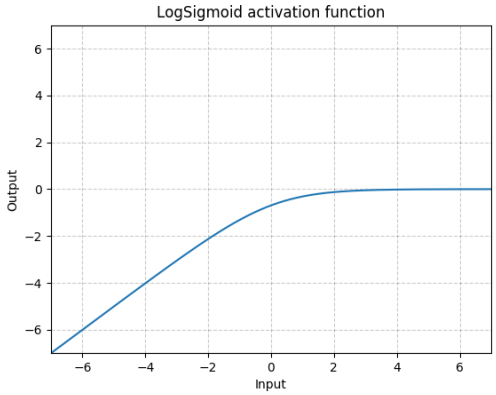

FIGURE 3.4: Logarithmic sigmoid

3.3 System evaluation

3.3.1 Loss function

As explained, when the final scores are converted into probabilities, the loss has to be calculated. In this case we use the Cross Entropy Loss, that includes the logarithmic softmax at the end of the network and the negative log likelihood loss (complete equation in Equation 3.3). It is useful to train a classification problem with C classes because it measures the performance of a classification model whose output is a probability.

$$Loss(x, class) = w[class] \cdot (-x[class] + log(\Sigma exp(x[j]))) \tag{3.3}$$

To have a fair representation of each class, we opted to use weights for each of them inversely proportional to its frequency in the whole dataset. These weights will be used to compute the loss.

3.3.2 Metrics

Following the balanced solution mentioned in subsection 3.1.1, the metric used for testing purposes is the balanced accuracy. This is useful for multiclass classification problems that have imbalanced datasets. It is defined as the average of recall obtained on each class.

$$\begin{aligned} &Accuracy(x, class) = \tfrac{1}{\Sigma \hat{w}[i]} \Sigma (x[class] == x[j]) \cdot \hat{w}[j] \\ &\text{where } \hat{w}[i] = \tfrac{w[i]}{\Sigma (x[class] == x[j]) \cdot \hat{w}[j]} \end{aligned} \tag{3.4}$$

Here, the weights are used as well.

3.4 System update

Back propagation is the method in which the parameters are updated before starting a new epoch to minimize the cost. The algorithm used is Stochastic Gradient Descent (SGD), that basically moves to the steepest direction following the cost function.

To the base SGD operations, we add two more concepts:

– Momentum, which remembers the velocity at each iteration, and updates its value as a linear combination of the gradient and the previous update.

$$v = \rho \cdot v + g,\ p = p - lr \cdot v$$
where v speed, p prediction, g gradient and lr learning rate (3.5)

– Weight decay / L2 regularization, which prevents the network from doing too well in the training (avoid overfitting).

$$g = (1 - lr \cdot \lambda) \cdot g$$
where g gradient, lr learning rate and λ regularization parameter (3.6)

3.5 Software and hardware specifications

The computers used to perform all the experiments were the server provided by GPI (gpu and cpu), a regular PC (just cpu) and Google Colab (gpu).

4

Experiments and results

4.1 Datasets

The main project dataset it consists of a large image taken the 2nd of July 2017 obtained with a Compact Airbone Spectrographic Imager (CASI) sensor. According to this sensor, the specifications for this image are 68 spectral bands and 50 cm of resolution. The image is 5108x7856 pixels, which makes 2.728.734.464 values for training and testing. However, only 52.280 are labelled manually. In Table 4.1 there is the class list with their respectives number of labelled samples.

As this project was thought to be not only result oriented, but also as a learning opportunity, another smaller dataset was previously used to understand the various architectures and techniques and to test which values were better. The Indian Pines dataset is widely used by papers dedicated to design DL architectures to learn segmentation for HSI (some of them mentioned in chapter State of the art). This dataset has 145x145 labelled pixels with 220 spectral bands, 20m resolution and 8 classes, showed in 4.2 (information from [7]). In Figure 4.1, we can see an image of Maspalomas park.

FIGURE 4.1: Maspalomas Park Image

#	Name	Samples
0	Unclassified	0
1	Trees	4216
2	Shrubs	4273
3	Sand	5623
4	Asphalt (roads)	3578
5	Grass	5548
6	Lake	4991
7	Sea	5020
8	Waves	1578
9	Others	3541
10	Swimming pools	4112
11	Built soil	4546
12	Rest bare soils	5254

TABLE 4.1: Maspalomas' class distribution

#	Name	Samples
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2455
12	Soybean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93

TABLE 4.2: Indian Pines' class distribution

4.2 Experiments with Indian Pines

The experiments with this dataset were extensive. The first major discovery was to cut the patches of the image in squares, not rectangular. For example, in figures 4.2 and 4.3 there is two example experiments made with U-Net and their respectives patch sizes.

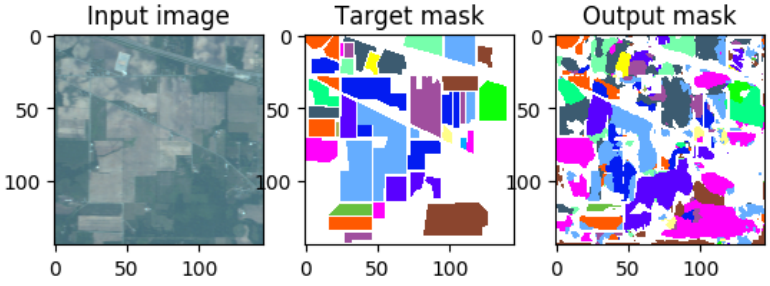

FIGURE 4.2: 32x16 patches

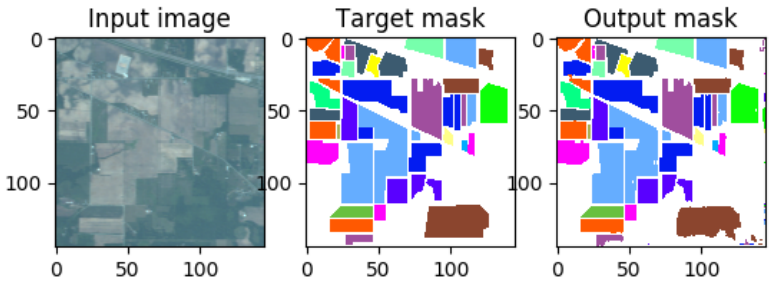

FIGURE 4.3: 32x32 patches

The best results at that point were the ones showing in Figure 4.4 and Figure 4.5. Both of them have almost the same configuration: 1000 epochs and a learning rate of 0.01. Batch size is the only difference: 30 for U-Net and 20 for 3D U-Net. The loss and accuracy are also below in Figure 4.6 and Figure 4.7.

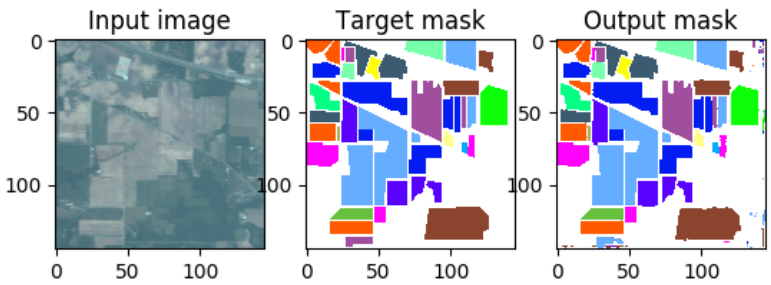

FIGURE 4.4: U-Net

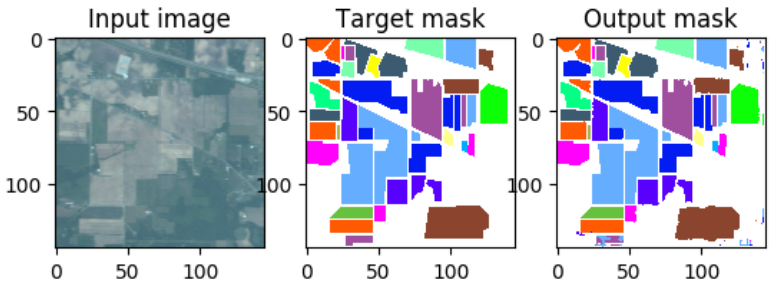

FIGURE 4.5: 3D U-Net

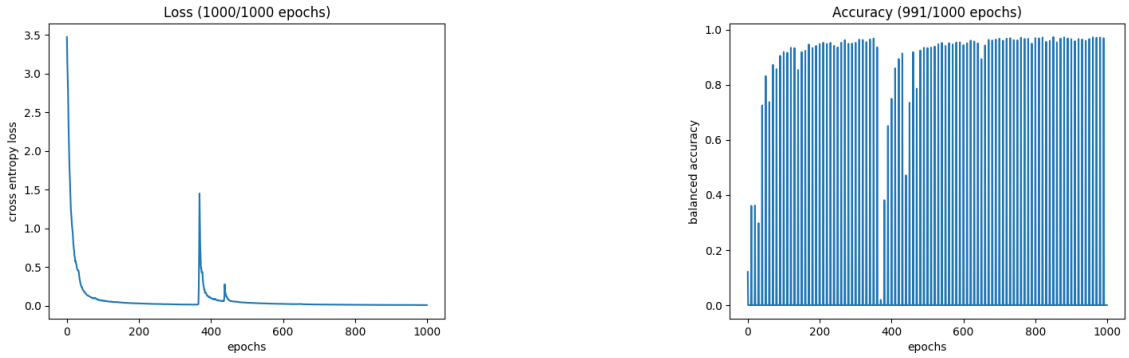

FIGURE 4.6: Loss and accuracy for U-Net (best results before balanced pixel distribution)

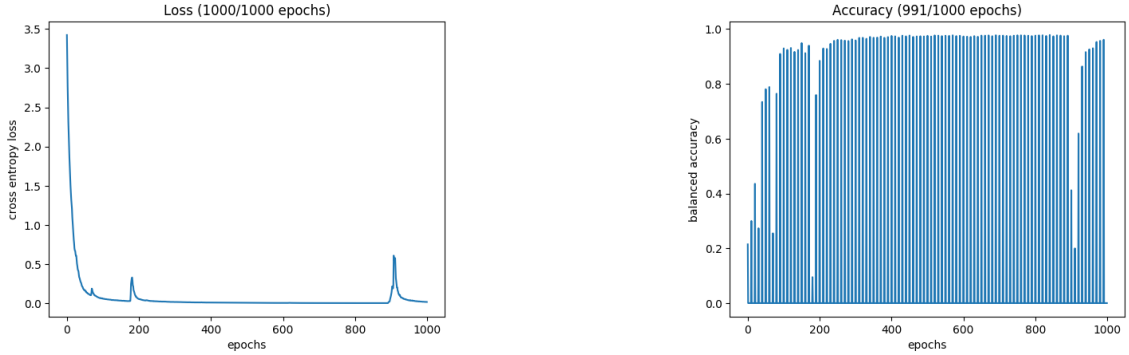

FIGURE 4.7: Loss and accuracy for 3D U-Net (best results before balanced pixel distribution)

The following change was to use stratified sampling and weighted measures. As explained in Methodology, the goal with this type of sampling was to equally divide training and validation pixels by classes; and then add weights to the calculations for loss and accuracy. This is important when the classes are unbalanced, because the network will just learn the majority class.

The percentages were 80% for training and 20% for validation. Before, the samples were divided randomly, without making sure how many were selected for training and validation to keep the same distribution as the original.

The configuration with better results is the same for both architectures: 1000 epochs, batch size of 30 and a learning rate of 0.01. See in Figure 4.8 and Figure 4.9 and in Figure 4.10 and Figure 4.11 the loss and accuracy results are represented.

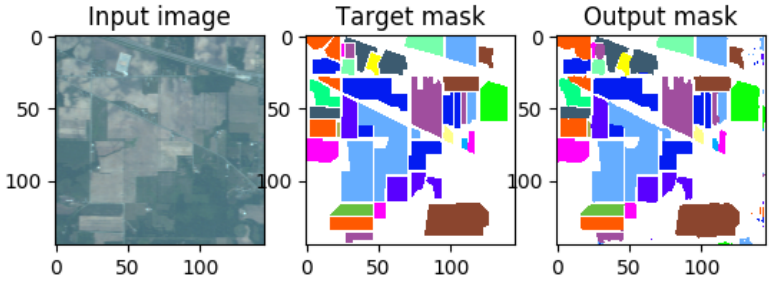

FIGURE 4.8: U-Net

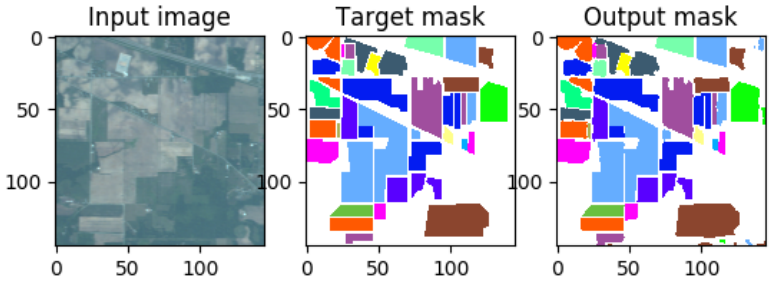

FIGURE 4.9: 3D U-Net

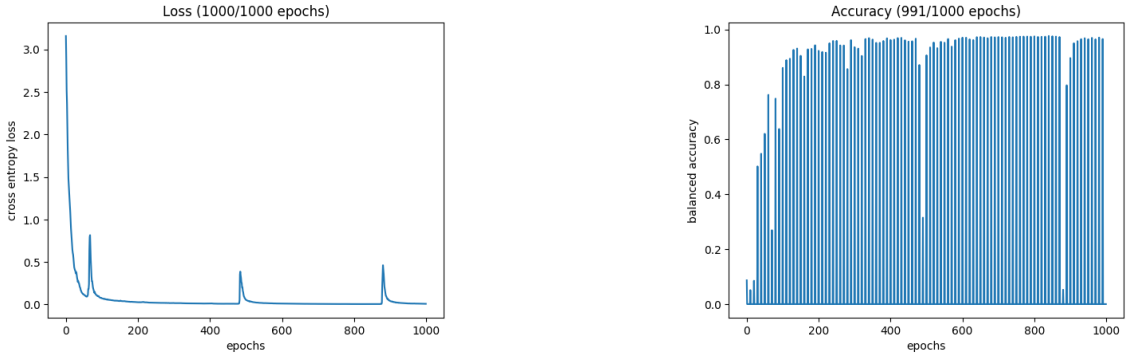

FIGURE 4.10: Loss and accuracy for U-Net (best results)

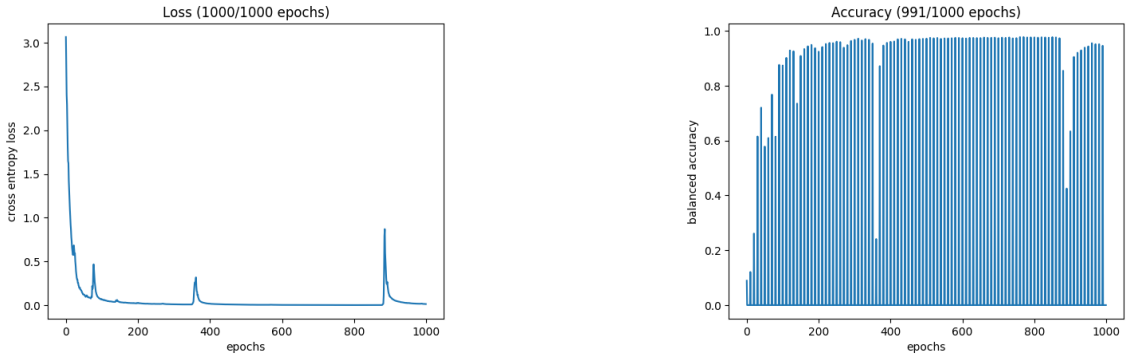

FIGURE 4.11: Loss and accuracy for 3D U-Net (best results)

The results for U-Net are better when we look at the numbers and at the prediction carefully. The fact that the dataset is very small, makes having less learnable parameters, as the U-Net has, better. Also, in this case, we were able to train for a lot of epochs to have the chance to see if the prediction would get more accurate or not, and the result is that the metrics are very close:

- In the last 500 epochs, the accuracy for the U-Net best experiment changes from 0.990772 to 0.993665 (difference of 0.002893)
- In the last 500 epochs, the accuracy for the 3D U-Net best experiment changes from 0.978909 to 0.986651 (difference of 0.007742)

These experiments lasted 4 hours more or less. For the two latter experiments, the percentage of correct labelling by class is in Table 4.3. The majority of values are bigger for U-Net, and that is because it has less parameters to learn in the same time (epochs).

#	Name	% for U-Net	% for 3D U-Net
1	Alfalfa	95.55	95.55
2	Corn-notill	97.70	97.62
3	Corn-mintill	91.13	95.27
4	Corn	92.31	89.75
5	Grass-pasture	95.18	92.16
6	Grass-trees	94.25	99.87
7	Grass-pasture-mowed	84.62	96.16
8	Hay-windrowed	89.98	91.01
9	Oats	95.00	95.00
10	Soybean-notill	97.11	96.18
11	Soybean-mintill	96.28	97.01
12	Soybean-clean	96.49	95.54
13	Wheat	96.7	95.76
14	Woods	99.02	96.44
15	Buildings-Grass-Trees-Drives	98.91	94.26
16	Stone-Steel-Towers	91.58	91.58

TABLE 4.3: Indian Pines' accuracy by classes

4.3 Experiments with Maspalomas

To start these experiments, two different points needed to be considered: first, remember that the manual labelled data was already divided into training and validating; secondly, the results for Indian Pines were analyzed. The goal was to have a result with low loss and high accuracy, but the number of epochs had to be reduced due to time constraints.
In this case, it is difficult to judge perceptually which configuration worked better, and that is due to two reasons:

- The ground truth is very scarce and, as we will see, the accuracy values may be optimum because the few labelled pixels are correctly predicted while the actual image doesn't look well-predicted.
- At the same time we are stating the latter, there is no way to know if the prediction looks correct or not, because there is no target for the whole image as there was in Indian Pines.
  What was done to compare results was to look at the SVM segmentation outcome by [12].

The more relevant experiments with this dataset are the ones that are mentioned in Table 4.4.

Architecture	Epochs	Batch size	LR	Final loss
U-Net	40	20	0.001	**0.02485**
3D U-Net	40	20	0.001	**0.03085**
U-Net	40	30	0.001	**0.04835**
3D U-Net	40	30	0.001	**0.05194**
U-Net	80	25	0.001	**0.00151**
3D U-Net	80	25	0.001	**0.00193**

TABLE 4.4: Relevant Maspalomas' experiments

To view these results, figures 4.12 and 4.13 show two patch examples, and Figure 4.14 its measures.

Note: Remember that all the target masks of the future figures are actually the result of SVM in [12]. Also, to see the colors corresponding to each class, see Appendix B: More Maspalomas' results.

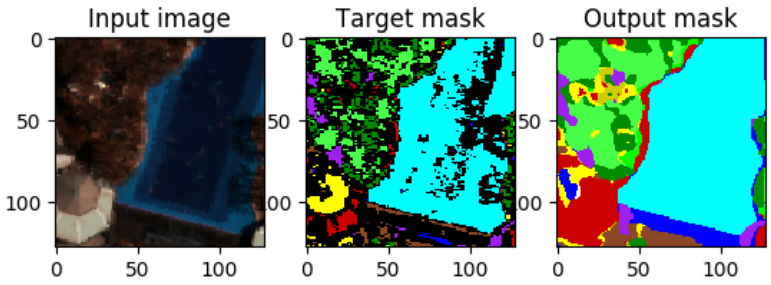

FIGURE 4.12: Example patch for first experiment in Table 4.4

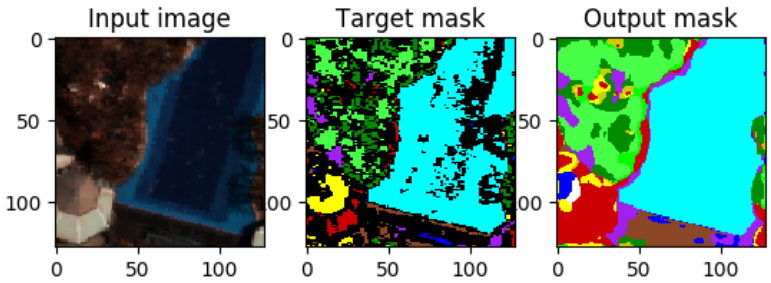

FIGURE 4.13: Example patch for second experiment in Table 4.4

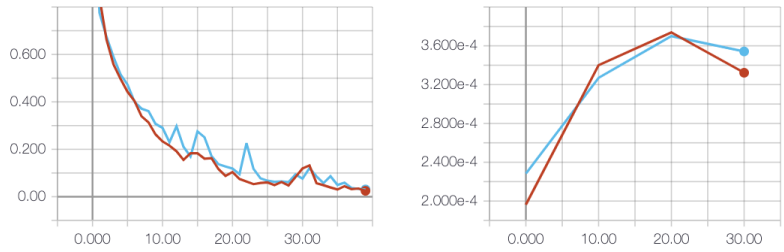

FIGURE 4.14: Loss and accuracy for first two experiments in Table 4.4

The fact that both of these examples output a validation accuracy of 1.0 proves the statements said before. The results for the third configuration (the two last ones of Table 4.4), in 4.15 and 4.16 there is an example patch and the measures in Figure 4.17. Another time, the accuracy for both of them is 1.0.

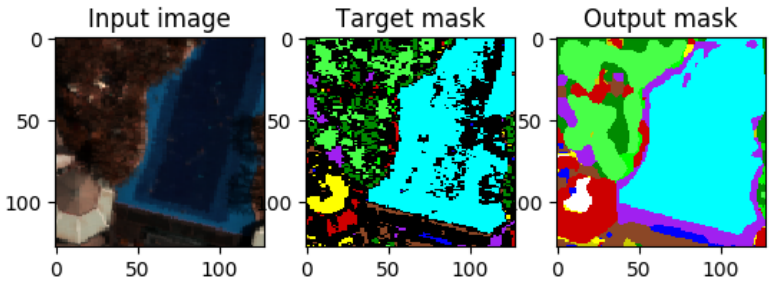

FIGURE 4.15: Example patch for second to last experiment in Table 4.4

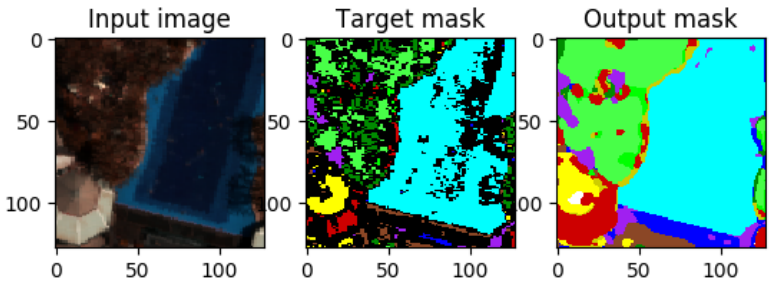

FIGURE 4.16: Example patch for last experiment in Table 4.4

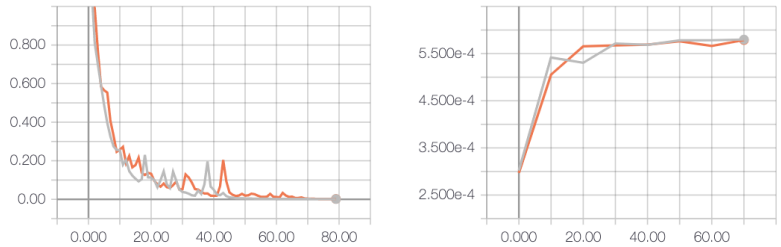

FIGURE 4.17: Loss and accuracy for last two experiments in Table 4.4

More results in Appendix B: More Maspalomas' results.

How will we value then the best parameters tested for U-Net and 3D U-Net in this dataset? For each configuration, we calculate the percentage of correct predicted labels for each class. The one with higher percentages will be the most accurate.

In Table 6.2, we see the best configurations' percentages. In both architectures, the best percentages are the ones with the model of 80 epochs. This result is not a surprise, as it has double the epochs from the other ones. However, it is interesting to see that the other ones do not perform badly. There are more low percentages, but not too many. The other results will be in Appendix B: More Maspalomas' results.

#	Name	% U-Net	% 3D U-Net
0	Unclassified	-	-
1	Trees	84.58	82.63
2	Shrubs	98.34	95.86
3	Sand	90.27	91.47
4	Asphalt (roads)	94.10	92.54
5	Grass	99.56	96.32
6	Lake	85.86	86.03
7	Sea	90.85	90.12
8	Waves	89.30	86.45
9	Others	97.89	95.74
10	Swimming pools	99.33	98.89
11	Built soil	99.46	95.93
12	Rest bare soils	62.30	65.24

TABLE 4.5: Maspalomas' accuracy by classes

5

Budget

Despite being a research project and therefore not involving a service or product to be sold, this section tries to estimate the budget of the project.

The hardware used for this project were the computational resources provided by the GPI and the personal computer used to develop and research. The total estimation of the hardware is the cost for the use of the server of approximately 50€a month and the computer approximately 40€a month (considering a computer of 2000€with a product life of 4 years).

The software used for the development and the visualization of large data is all open-source. However, the software used to produce the SVM segmentation was not. The annual licence fee is approximately 220€.

Finally, the salary of the members involved in it. Considering the amount of time that each member has put into this project and the standard salary for junior engineers (15€/hour, 15 hours a week), senior engineers (20€/hour, 5 hours a week) and technical advisors (30€/hour, 2 hours a week); the costs can be summarized as follows:

Item	Price	Time	Total (4 months)
Server computation	50€/month	*all*	200€
Computer	40€/month	*all*	160€
ENVI Software	220€/year	*all*	220€
Junior engineer	15€/hour	15 hours/week	3600€
Senior engineer	20€/hour	5 hours/week	1600€
Advisor	30€/hour	3 hours/week	1440€

TABLE 5.1: Budget

This makes a total of:

$$200 + 160 + 220 + 3600 + 1600 + 1440 = \mathbf{7220€} \tag{5.1}$$

6

Conclusions and future development

As for the Indian Pines dataset, the results were successful. The architecture was simple and the dataset very small. The latter causes overfitting.

This work was mostly experimental and the results for Maspalomas were not optimal. However, this leaves more room for improvement. If we refer to Maspalomas, the ground truth was not extensive at all. Some of the patches did not have labelled data and the majority had not many (less than 500 mostly), but the compromise to get a big enough patch to have relevant information and the memory restriction affected the size decision. This fact made the accuracy-loss values during training not representative enough. This means that, because of the difference being performed with a few pixels of a patch (the maximum was less than 7% in comparison to the total number of pixels), the inequality appeared higher than what it seamed.

This incongruity can be seen when comparing the results with the SVM result of [12]: when looking at the summary table with the worst and best classified classes, there are percentages close to the results obtained with our proposed configurations. Nonetheless, the predicted patches look different in the sense that their segmentation looks more precise.

The option of manually labelling more data or maybe using the result of [12] as ground truth to see how the experiments resulted could be immensely beneficial to get a better outcome.

Technically speaking, more hyperparameters that were limited due time constraints could be modified in order to see how it would perform with more time, for example the number of epochs could have maybe been higher. Specially with the 3D U-Net, as it has more parameters to learn.

Bibliography

[1] URL: https://pytorch.org/.

[2] Fahim Irfan Alam et al. "Conditional random field and deep feature learning for hyperspectral image classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 57.3 (2018), pp. 1612–1628.

[3] Tom Brosch et al. "Deep convolutional encoder networks for multiple sclerosis lesion segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 3–11.

[4] Özgün Çiçek et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[5] Qishuo Gao, Samsung Lim, and Xiuping Jia. "Hyperspectral image classification using convolutional neural networks and multiple feature learning". In: *Remote Sensing* 10.2 (2018), p. 299.

[6] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[7] Grupo de Inteligencia Computacional. *Hyperspectral Remote Sensing Scenes*. 2014. URL: http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[9] Hyungtae Lee and Heesung Kwon. "Going deeper with contextual CNN for hyperspectral image classification". In: *IEEE Transactions on Image Processing* 26.10 (2017), pp. 4843–4855.

[10] Milesial. *Pytorch-UNet*. 2007. URL: https://github.com/milesial/Pytorch-UNet.

[11] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation". In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 565–571.

[12] Juan Daniel Moreno Gázquez. "Clasificación de imágenes multiespectrales e hiperespectrales de alta resolución para la obtención de cartografia temática en Maspalomas". In: (2018).

[13] Arkadiusz Nowaczynski. *Deep learning for satellite imagery via image segmentation*. 2017. URL: https://deepsense.ai/deep-learning-for-satellite-imagery-via-image-segmentation/.

[14] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[15] Swalpa Kumar Roy et al. "HybridSN: Exploring 3D-2D CNN Feature Hierarchy for Hyperspectral Image Classification". In: *arXiv preprint arXiv:1902.06701* (2019).

[16] Jamie Sherrah. "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery". In: *arXiv preprint arXiv:1606.02585* (2016).

[17] M. Mitchell Waldrop. "News Feature: What are the limits of deep learning?" In: *Proceedings of the National Academy of Sciences* 116.4 (2019), pp. 1074–1077. ISSN: 0027-8424. DOI: 10.1073/pnas.1821594116. eprint: https://www.pnas.org/content/116/4/1074.full.pdf. URL: https://www.pnas.org/content/116/4/1074.

[18] Xiaofei Yang et al. "Hyperspectral image classification with deep learning models". In: *IEEE Transactions on Geoscience and Remote Sensing* 56.9 (2018), pp. 5408–5423.

Appendix A: Organization

The initial work structure was the following:

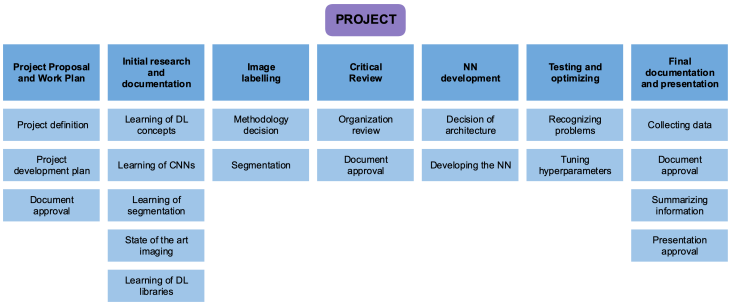

FIGURE 6.1: Initial work structure

These are in order of Work Package, so from left to right we have WP1 to WP7.

Appendix B: More Maspalomas' results

The classes with the corresponding color:

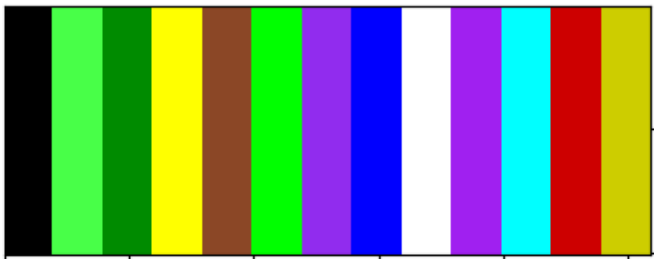

FIGURE 6.2: In order, the classes are: Unclassified, Trees, Shrubs, Sand, Asphalt (roads), Grass, Lake, Sea, Waves, Others, Swimming pools, Built soil, Rest bare soils

Another result (U-Net / 3D U-Net with 30 epochs, 30 batch size and 0.001 learning rate):

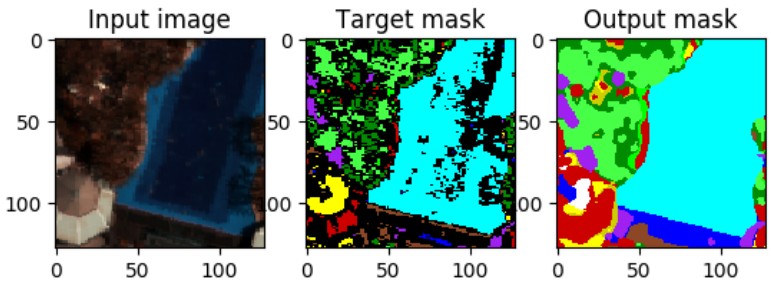

FIGURE 6.3: Example patch for third experiment in Table 4.4

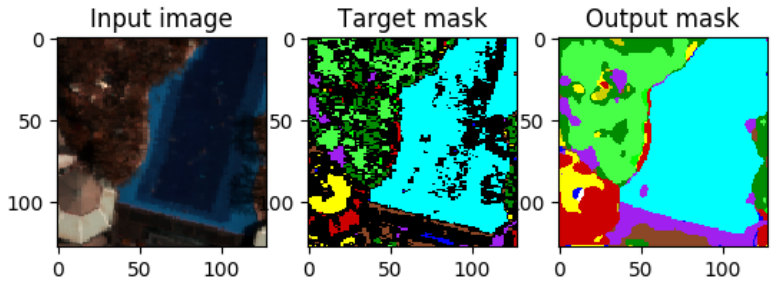

FIGURE 6.4: Example patch for forth experiment in Table 4.4

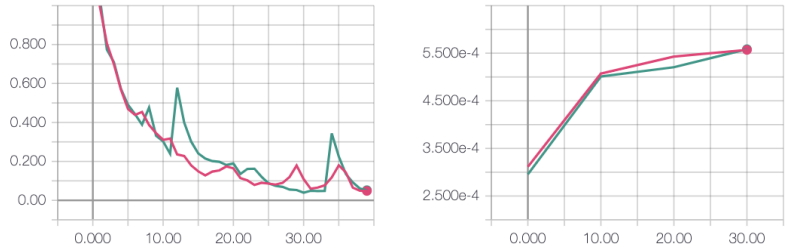

FIGURE 6.5: Loss and accuracy for third and fourth experiments in Table 4.4

Here, the other accuracy percentages for Maspalomas:

#	Name	% U-Net	% 3D U-Net
0	Unclassified	-	-
1	Trees	76.26	79.38
2	Shrubs	98.30	97.58
3	Sand	92.95	83.22
4	Asphalt (roads)	94.06	74.97
5	Grass	99.63	99.98
6	Lake	68.12	86.12
7	Sea	87.15	88.01
8	Waves	86.60	81.86
9	Others	93.17	99.21
10	Swimming pools	99.52	99.87
11	Built soil	99.81	94.63
12	Rest bare soils	67.65	78.34

TABLE 6.1: Maspalomas' accuracy by classes for 40 epochs, a batch size of 20 and a learning rate of 0.001

#	Name	% U-Net	% 3D U-Net
0	Unclassified	-	-
1	Trees	77.44	74.53
2	Shrubs	97.33	98.01
3	Sand	82.39	80.67
4	Asphalt (roads)	69.14	65.82
5	Grass	100.00	99.54
6	Lake	88.44	85.98
7	Sea	99.20	92.13
8	Waves	94.05	91.77
9	Others	99.18	99.34
10	Swimming pools	98.18	97.20
11	Built soil	95.07	93.87
12	Rest bare soils	59.18	63.07

TABLE 6.2: Maspalomas' accuracy by classes for 40 epochs, a batch size of 30 and a learning rate of 0.001